How Xyz Salvaged Client Data from an Older Version of a Software Application
A Term Paper in Decision Sciences, Executive MBA, American University in Bulgaria

I. Task scope and background information

1. Task scope: consider a business or personal decision of which you have first hand knowledge. It must involve choosing between two or more distinct alternatives, not just "yes or no." Write three separate essays: one on the intelligence phase, one on the design phase, and one on the choice phase. Each separate essay must be at least two pages (double spaced) and at most five pages. Depending on the decision you choose, some of your essays might be much longer than others.

2. Background information: I work for a software development company called Xyz. The essays below are about a business situation with a Xyz-built software application called Dynamo. A brief description of the application will put the situation in context.

Dynamo is a client-server application. Data is stored centrally on a server, clients - a client here is synonymous to a licensed user who has the software installed - connect to the server to retrieve the latest data and save any additions or edits they have made.

When a secure connection to the server is not available Dynamo can work in offline mode. This feature is used by people who travel often. When offline, a user will access a copy of the data saved on the computer they carry with them. Once in a while, when a secure connection can be established, the offline data on the computer is synchronized with the online data on the server.

When a new version is released, we replace the software on client computers and migrate the data on the server from the old to the new version. Due to technological limitations, there is no backward compatibility: ver. X+1 of the software does not work with ver. X of the data and vice versa. The data - or rather the database structure - and the software must
have the same version.
II. Essays

1. Intelligence Phase

1.1. How did you or your organization become aware that the decision needed to be made?

A week after we upgraded a customer who owns 25 user licenses to the latest Dynamo version we were alerted that a single user based in a far-away branch office cannot synchronize her offline data with the online data on the server. We found that first, the user still had the previous version which meant that the network consultant who should have replaced the software had been negligent, and second, that the user had worked offline for about 3 months, input a lot of data, but never synchronized. The new software was already deployed company-wide, the online data was migrated. The offline data the user had been updating was incompatible with the new software. The decisions that I had to make at this stage were what to do about the legacy data and how to prevent similar surprises in the future.

I was facing a disturbance problem, a gap between the intended use of a feature in our software (offline mode) and its actual use -

- How the feature should be used: when you work offline, you have to synchronize every 2 weeks to get the new data from the server and let all other online users see the data you have added or modified while offline.
- How the feature had been used: a user had worked offline for months without synchronizing; the online users had not noticed they do not see the data input by the offline user; the offline user had not noticed she does not see the new data input by the 20+ online users.

The mental model of the offline user differed from the mental model of our application designers. For security reasons, the latter had designed the application to require a user action in order to synchronize: laptops get stolen, people change jobs, and the data in question is sensitive. The user’s mental model was that Dynamo silently and automatically synchronizes her data on a regular basis.
1.2. What information resources were already available to support the decision?

A lot of information was available: I was familiar with this customer’s support history, I had met the key stakeholders in person and had been in touch with them often via email and phone. I knew they had had offline users in the past, though at the time of the upgrade I was ensured that no one worked offline anymore. I had our product documentation that describes what our software can do - and how - plus what the software is designed to do; I knew how much I can bend the design-and-usage rules without doing serious damage. I had sufficient industry knowledge - our clients are venture capitalists and investment funds - to understand why offline mode was designed in this way. I had access to third-party product documentation about the software and technologies our application relies on, I understood why we do upgrades in this way and why we do not offer backward compatibility. I had access to the system architects who designed the application and to the product developers who wrote the programming code.

1.3. What information resources if any were obtained specially to support the decision?

I obtained a copy of the offline data. The user claimed she had made a lot of new entries but there was no evidence for that. The null hypothesis was that no new entries had been made and it was testable: I had my engineers compare the offline to the online data. (In databases, data is stored in tables so we ran queries that returned the total number of items in a table and the date on which the table was last modified). This is how I learned the offline data was worth salvaging. I obtained valuable information that had the potential to change my actions.

I had my engineers run the standard upgrade tools - that we use to migrate data from one version to the next - over the offline data, then attempt a two-way synchronization between the migrated offline data and an online database of the same version. Our upgrade tools performed well but the resultant data could not be synchronized because of a third-party software limitation we were not aware of until then. This is how I learned there was no quick-and-dirty solution that could salvage the offline data.
1.4. In hindsight, how might the intelligence phase have been performed better?

Honestly, I do not know. My people are well-trained and efficient, I knew what we were up against. The only improvement I can think of is skipping the second test in 1.3 above because: a) ideally, we could have deduced it would not work before we spent half a day to try it or b) by sheer luck, we could have had stumbled across the same third-party limitation in the past. If any of these were true, I would have canceled the test with the standard upgrade tools and saved half a day.

1.5. What cognitive heuristics were helpful to you or your group in the intelligence phase?

*Prudence* was helpful. While we were striving to identify the scope of the problem, we tried hard not to adopt a we-know-it-all attitude. Practice had taught us to be humble. On multiple occasions in the past we had seen our good knowledge of our software - and of human-computer interaction in general - lead us to wrong conclusions just because we could not figure out the next unintended use of a feature or the next unconscious abuse of a documented limitation that happen when people use software. We had learned to ask questions first, confirm that the real-life incident supports any assumptions we made, and then speak up to offer a solution.

1.6. What cognitive biases were problems to you or your group in the intelligence phase?

The *status quo* was a problem, though strictly speaking this bias was not limited to the intelligence phase but turned out to be a big-picture item spanning over the entire problem solving process. Offline mode had been the wrong choice for this user from the very start. She was not working on a laptop and never traveled. Years ago, another manager had suggested offline mode to the user, then at subsequent upgrades - when we should have found out it is a bad fit - we had always assumed that since the user worked in offline mode in the previous version, she must be set up in offline mode in the next version.

When I evaluated the status quo as if it were not and discussed it with people who had not been involved in the original decision, we were able to see clearly that offline mode was not
the best option for this user. I wrote a recommendation memo to a senior manager with our customer and advised the discontinuation of offline mode usage for users who do not work on the road.
2. Design Phase

2.1. How did you or your organization come up the alternatives you considered?

I wish I could say that we started with the end in mind but we did not. If we had, we would have focused on the task at hand: all the users had to see all the data. All users equals online users plus offline users. All the data equals online data plus the new entries in the offline data.

One method we considered was to go back to the partial solution in II.1.3., then work around the third-party limitation by means of a huge re-programming effort in their system. The result from this method would have been rather uncertain, which is always the case when you edit or build upon somebody else’s programming code.

Another method was to roll back the new version, synchronize the offline data with the online data - that would have had the same version after the rollback - then re-do the upgrade to the new version. This would have been expensive, time consuming, and damaging to our reputation for technical excellence. This method would guarantee success with no degree of uncertainty whatsoever but we never considered it seriously.

Yet another alternative was to do nothing but explain to the customer how the situation came to be and advise them to re-input all the offline data that we could not move to the online data. I am proud to say this was a hypothetical option that no one even mentioned. We did not get where we are by turning a blind eye to a customer in distress.

2.2. How creative was the process?

At the time, I did not think about creativity. Now, I would say it must have been creative enough.

The creativity enhancing technique we unknowingly applied was lateral thinking. To find a good solution, we changed the scope of the problem. Our task was not to upgrade the offline database to the new version or to synchronize the data.
The real task was to take the new entries from the offline data in the old version and move them to the online data in the latest version. Synchronization was not required, tweaks to somebody else’s programming were not required either.

2.3. In hindsight, how might the design phase have been performed better?

The design phase might have been performed much better, if I had viewed the problem as an ordinary user outside Xyz, as opposed to an insider. I would have known that all I need is to get those new entries from the offline data that I have not got.

Instead of thinking about what people need, I thought about what Xyz can do. When you start on this track - and have some of the smartest programmers in the country who can do almost anything - you will never end up with a short list of possible solutions and most of the solutions on your list will not be simple. They will be brilliant, they will make use of state-of-the art technology, but are likely to be hopelessly over-engineered.

2.4. What cognitive heuristics were helpful to you or your group in the design phase?

Neutral framing was helpful. As we discussed and put down possible solutions, I never asked the group how we can make this or that work. I deliberately asked will it work. Despite the fact that I have a low tolerance for yes-men and tend to encourage conflict, I did not want to frame the problem in any way. As the consultative-autocrat-in-residence, I needed objective and frank responses, so I made sure my framing was as neutral as possible.

2.5. What cognitive biases were problems to you or your group in the design phase?

My selective perception bias created a big problem for the group. I was the only group member with a lot of customer exposure and the only team member with no background in computer science or engineering. My background is in the arts, I hold a master’s degree in literature. In a technology company, a non-technical viewpoint is an enormous asset that I had but did not use this time. Instead, I was blinded by role - a team leader whose team
faces a difficult business situation - and expertise. I was too familiar with the development process, too well aware what goes on behind the scenes on the programming level when people expand menus, select options, and click buttons in the user interface.

*Anchoring* was a another big problem for the group. We gave disproportionate weight to the first information we received which was that a user cannot synchronize. If we had followed Hammond’s recommendations on avoiding anchors - i.e. viewed the problem from different perspectives, tried to use alternative starting points - we would have saved a lot of time. We removed synchronization from the scope of the problem early enough but we should have also removed it from every possible solution or even better - refused to consider any solution that included or required synchronization.
3. Choice Phase

3.1. How did you or your organization decide which of the alternatives to implement?

The simple solution was data import: identify the new entries in the offline data from the old version, then import them into the online data in the latest version. We do this every time when our sales force wins a deal from a competitor - customers give us the data from the competitor’s system, we do the necessary data manipulations to align the data to our system, then import it. The process is automated, though it requires some additional programming because various data structures exist.

We chose the simplest solution that worked at a reasonably low cost in terms of man-hours.

3.2. Did you address the issue of uncertainty or probability?

I did not need to address uncertainly or probability. The problem I was working on was not numerical. Comparing the expected outcomes of the dozen solutions considered within the group would not have yielded any valuable information.

Most of the solutions were readily testable, so by the time I had to choose one, I would have known which had succeeded and which had failed. My short list would have contained only alternatives that succeeded with certainty which is better than any untested alternative whose probability of success is 1.

3.3. In hindsight, how might the choice phase have been performed better?

Same as in II.1.4, it is hard to say.

The conceptual breakthrough came in the design phase, it is outlined in II.2.3 above. After the paradigm shift, the best choice became obvious, a fact that shortened and simplified the choice phase.

3.4. What cognitive heuristics were helpful to you or your group in the choice phase?
See my answer to III.3.3 above. The choice phase was short and its output was very much predetermined.

3.5. What cognitive biases were problems to you or your group in the choice phase?

Here, the *selective perception bias* created a problem for another group member. The combination of role (senior product developer position) and expertise (in-depth knowledge of the underlying technology) lead to a partial failure to grasp the complexity of the situation. However, it should be noted that this particular group member was not involved in the intelligence phase, so what I list here as a selective perception bias could also be interpreted as a perceptual block to creativity - an inability to see the problem through various viewpoints that was necessary to grasp its complexity - a block that would not be there if that person knew everything that the group already knew.

**III. References**


